Distributed Systems

COMP90015 2019 Semester 1
Tutorial 04

Questions

Q1. Briefly discuss three aspects of the Socket interface.

Q2. Briefly explain three possible failures that can happen when using UDP for communication.

Q3. Briefly explain three aspects of TCP that address issues not addressed by UDP.

Q4. List the steps involved at the client and at the server to establish a TCP stream socket connection.

Questions

Q5. Give a reason why representing data using XML is preferable over a scheme such as CORBA’s data representation.

Q6. Give a reason why JSON data representation is preferable over XML.

Sockets

Q1. Briefly discuss three aspects of the Socket interface.

- Bound to a local port.
- Socket can be used for sending and receiving.
- Each socket is associated with a protocol (UDP or TCP).

UDP vs TCP

UDP: User Datagram Protocol

- Provides a message passing abstraction.
- Is the simplest form of Interprocess Communication (IPC).
- Transmits a single message (called a datagram) to the receiving process.

TCP: Transmission Control Protocol

- Provides an abstraction for a two-way stream.
- Streams do not have message boundaries.
- Stream provide the basis for producer/consumer communication.
- Data sent by the producer are queued until the consumer is ready to receive them.
- The consumer must wait when no data is available.

UDP vs TCP

Q2. Briefly explain three possible failures that can happen when using UDP for communication.

UDP vs TCP

Q2. Briefly explain three possible failures that can happen when using UDP for communication.

- Data Corruption.
- Omission failures.
- Order.

UDP vs TCP

Q3. Briefly explain three aspects of TCP that address issues not addressed by UDP.

UDP vs TCP

Q3. Briefly explain three aspects of TCP that address issues not addressed by UDP.

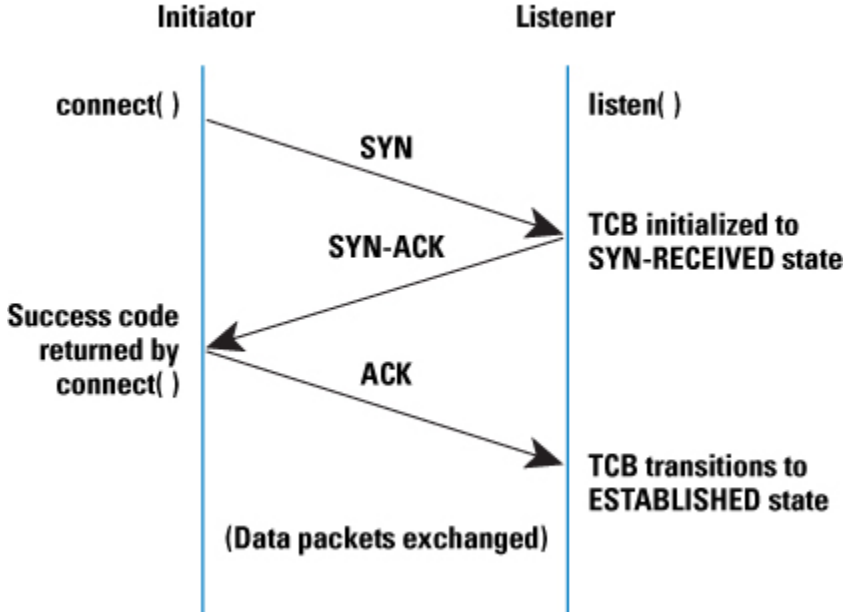
- **Message sizes:** There is no limit on data size applications can use.
- **Lost messages:** TCP uses an acknowledgment scheme unlike UDP. If acknowledgments are not received the messages are retransmitted.
- **Flow control:** TCP protocol attempts to match the speed of the process that reads the message and writes to the stream.
- **Message duplication or ordering:** Message identifiers are associated with IP packets to enable the recipient to detect and reject duplicates and reorder messages in case messages arrive out of order.
- **Message destinations:** The communicating processes establish a connection before communicating. The connection involves a connect request from the client to the server followed by an accept request from the server to the client.

UDP vs TCP

Q4. List the steps involved at the client and at the server to establish a TCP stream socket connection.

UDP vs TCP

Q4. List the steps involved at the client and at the server to establish a TCP stream socket connection.



Recap Course Materials

- Why do we need external data representation and marshalling?
  - For data transmission: data structures in programs are flattened to a sequence of bytes before transmission
  - To mask heterogeneity in data representations: ASCII vs Unicode
- What is External data representation?
  - Agreed standard for representing data structures and primitive data
- What is Marshalling and Unmarshalling?
  - Process of converting/ disassembling the data to/from the form suitable for transmission
- What are the common approaches for external data representation?
  - CORBA’s Common Data Representation (CDR)
  - Java's object serialization
  - Extensible markup language (XML)
  - JSON (JavaScript Object Notation)

Recap Course Materials

- Java Object Serialization
  - Serialization refers to the activity of flattening an object to be suitable for storage or transmission
  - Deserialization refers to the activity of restoring the state of the object
- What information is being serialized?
  - Information about the class of the object: class name, version, etc.
  - All objects it references are serialized as handles
  - Contents of primitive instance variables that are primitive types
- How to serialize a object of a user class?
  - Implement the Java Serializable interface (Opposite: transient)
  - Contain a *private static final long* variable named *serialVersionUID*.

Interprocess Communication

Q5. Give a reason why representing data using XML is preferable over a scheme such as CORBA’s data representation.

Interprocess Communication

Q5. Give a reason why representing data using XML is preferable over a scheme such as CORBA’s data representation.

- XML is self describing unlike CORBA CDR.
  - Tags describe the logical structure of the content
  - Tags are generic - unlike HTML where tags give display instructions
- XML is extensible
  - Additional tags can be defined later
- Human-readable
  - Tags together with namespaces allow the tags to be meaningful
  - Data is textual, it can be read by humans and different platforms

CORBA Example

index in sequence of bytes	← 4 bytes →	*notes on representation*
0–3	5	*length of string*
4–7	"Smit"	*'Smith'*
8–11	"h___"	
12–15	6	*length of string*
16–19	"Lond"	*'London'*
20-23	"on__"	
24–27	1934	*unsigned long*

The flattened form represents a *Person* struct with value: {'Smith', 'London', 1934}

XML Example

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Attribute

Element

Element

Interprocess Communication

Q6. Give a reason why JSON data representation is preferable over XML.

JSON:

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML:

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Interprocess Communication

Q6. Give a reason why JSON data representation is preferable over XML.

- JSON requires less configuration overhead --- it's easier to program for reading and writing.
- XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function into a into a ready-to-use JavaScript object.
- JSON serialization produces shorter strings than XML. Using JSON will reduce the amount of data transmission and improve performance

Example: Twitter.

Project Discussion

End of Tutorial